Cont A2 27. (Amended) The method as claimed in claim 4 wherein the status of an object in the thread heap is changed to global if the object is assigned to a static variable or if the object is assigned to a field in a global object.

REMARKS/ARGUMENTS

This communication is responsive to a non-final Office Action dated August 27, 2002 rejecting claims 1-27 of the instant patent application (hereafter the "Office Action"). Therefore, claims 1-27 have been amended. Reconsideration of the application is respectfully requested.

I. Drawings

The Office Action objected to the drawings under 37 CFR § 1.83(a) for failing to show step 4.8 of Figure 4 as described in the specification on page 14, line 20. Accordingly, a copy of Figure 4 showing a proposed amendment is attached. Therefore, the objection has been overcome.

II. Claim Rejections – 35 USC § 112

The Office Action rejected claim 24 under 35 USC § 112, first paragraph, as "containing subject matter which was not described in the specification in such a way as to enable one skilled in the art to which it pertains, or with which it is most nearly connected, to make and/or use the invention." Applicant respectfully traverses the rejection on grounds that the specification does indeed disclose the claimed means. The claimed means for deleting covers the hardware and logic embodied in the platform 10 as shown in Fig. 1 and discussed at page 10 of the specification. Moreover, please note that a deletion function is expressly discussed at page 3 of the specification. Deletion of objects in a garbage sweep is known in the art. The invention claimed herein includes a deletion function in combination with other steps, means or instructions.

The Office Action rejected claim 6 under 35 USC § 112, second paragraph, as being indefinite for failing to point out whether the thread heap is the same or a different thread heap from the thread heap of claim 1. Claim 6 has been amended to overcome this rejection.

III. Claim Rejections – 35 USC §102(b)

Claims 1-7, 14, 16, 17, and 22-27 have been rejected under 35 U.S.C. §102(b) as being anticipated by Kolodner (US Pat. No. 6,289,360). This rejection is overcome because it is believed that claims 1-7, 14, 16, and 22-27, as amended, are not anticipated by Kolodner. Nowhere does Kolodner teach or disclose monitoring objects as required by claims 1-7, 14, 16, and 22-27. Instead, Kolodner discusses tracing objects from the stack and then tracing references from within those objects.

Claim 1 as amended requires the step of monitoring the object to determine whether the object is referenced only from a given thread stack. The amendment was made to clarify what is meant by a "local root" in this context. Kolodner does not discuss monitoring each object that is local to the given thread to determine whether the object is referenced only from a given thread stack as required by claim 1 and its dependent claims. The part of Kolodner cited in the Office Action (col. 2, lines 5-15) discusses identification of live objects. The cited part only mentions the global state and local state of each thread, and does not concern monitoring objects to determine determining whether an object is referenced only from a given thread stack, as claimed. Therefore, Kolodner does not anticipate claim 1 or any of its dependent claims.

Claim 2 is further distinguishable from Kolodner because Kolodner neither teaches nor suggests assigning a local status to the monitored objects which status is changed under certain conditions. The part of Kolodner cited by the Office Action relates to three synchronization statuses for a mutator thread. Claim 2 concerns assigning a local status to an object. The discussion cited in Kolodner has nothing to do with the subject matter of claim 2 other than the coincidental use of the word "status."

Claim 3 recites the step of deleting from the thread heap one or more local objects when it is determined that they are not accessible from a local root. The Office Action cited col. 2, lines 8-11 of Kolodner. As discussed above, section discusses the determination of whether objects are "live." It does not relate to deletion of objects that are determined to be not accessible from a local root.

Claim 4, as amended, requires changing the object status to global when it is determined that the object is accessible from either a local root. The Office Action cited col. 8, lines 40-53 and col. 11, lines 46-47 of Kolodner. As stated above, Kolodner does not teach or suggest monitoring the local/global status of objects.

Kolodner does not anticipate claims 5-7, 14, 16, 17, and 22-27 for at least the same reasons as discussed above.

III. Claim Rejections – 35 USC §103(a).

The Office Action rejected claims 8, 9, and 15 as obvious over the Kolodner '360 patent in view of US Patent 5,966,702 to Fresko.

The Fresko patent deals with grouping class files together in single combined multi-class file (called a jar file in Java terminology). In particular the patent deals with the pre-processing and packaging of the class files. The Kolodner patent deals with a mechanism for reducing/eliminating the synchronization between sweep and allocation for a concurrent garbage collector.

Claims 8, 9, and 15 each recite elements that are patentable in combination with their respective base claims. For reasons discussed above, Kolodner neither teaches nor suggests the base claims upon which claims 8, 9, and 15 depend. Since the Office Action concedes that the limitations of claims 8, 9, and 15 are not disclosed by Fresko and Fresko does not relate to the problems concerning the claimed subject matter we must conclude that the combination of Kolodner and Fresko would not have rendered the claims in issue obvious.

In view of the remarks made in support of the base claims, the additional elements recited in claims 8, 9, and 15 are patentable in combination with the base claims.

The Office Action rejected claims 10-13 as obvious over Kolodner in view of Fresko and further in view of O'Connor et al (US Patent No. 5,953,736).

Regarding claim 10, O'Connor discusses a 32-bit word with additional storage reserved for synchronization status of the object. However, O'Connor neither teaches nor suggests the use of spare capacity in an object header for status of the object.

Regarding claim 11, as discussed above with respect to claim 10, O'Connor does not discuss using the spare space available in the header portion as the flag, or status of the object. Moreover, since O'Connor does not at all relate to monitoring the object to determine if it is referenced only from a given thread stack, there is no suggestion, motivation, or teaching that would have motivated one skilled in the art to combine the references.

Regarding claim 12, Fresko does not disclose in its Abstract moving objects whose status is global from the thread heap to a global heap. In fact, Fresko does not discuss global objects or global heaps in its Abstract.

The Office Action rejected claims 18-21 as being obvious over Kolodner in view of Jagannathan et al. (US Patent No. 5692193). Jagannathan relates to software architecture for control of highly parallel computer systems. Jagannathan is cited for teaching the local thread stacks and heaps and a global heap. Applicant respectfully submits that the claimed respective local thread stacks and heaps, and a global heap must be viewed as part of the claimed combination and that absent some evidence of a teaching, suggestion or motivation the mere presence in a prior reference of elements having similar or same names is not a proper basis for a conclusion of obviousness. Moreover, even if they were properly combinable, the cited references still fail to teach or suggest the claimed means for monitoring each object that is local to the given thread, to determine whether the object is accessible from any thread other than the given thread. Claims 22-27 are patentable for the foregoing reasons. Therefore, claims 18-27 would not have been obvious to those skilled in art in view of the cited references.

Attached hereto is a marked-up version of the changes made to the specifications and claims by the current amendment. The attached page is captioned **"Version with markings to show changes made."**

Applicant respectfully requests that a timely Notice of Allowance be issued in this case.

Respectfully submitted,

Please send correspondence to:
Michael J. Buchenhorner, P.A.
1430 Sorolla Avenue
Coral Gables, Florida 33134

By: Michael J. Buchenhorner

Michael J. Buchenhorner
Registration No. 33,162
Telephone No.: (305) 529-0221

VERSION WITH MARKINGS TO SHOW CHANGES MADE

In the specification:

Please replace the paragraph beginning at page 14 line 6 in the specification with the following:

--The process of memory management is described with reference to Figure 4. In Step 4.1 an object is created from a class stored in the class area 36. Step 4.2 checks to see if the class is a global class, i.e., a class all of whose instances are global or one whose instances we expect to quickly become global, whereby the object is assigned to be global and/or placed in the global heap (Step 4.7). Step 4.3 calculates the size of the object [is calculated] to determine [see] whether it will fit in the thread heap. Step 4.4 [if there is not enough space then garbage collection is performed] performs garbage collection to free up memory. Step 4.5 places the object in the thread heap memory along with the object length in multiples of eight. Step 4.6 uses a spare bit in the length attribute as a flag and sets it as local ('0'). The process ends at Step 4.8.--

In the claims:

1. (Amended) A method of managing memory in a multi-threaded processing environment including respective local thread stacks and heaps and a global heap, said method comprising:

creating an object in a thread heap; and

monitoring the object to determine whether the object is [a local root] referenced only from a given thread stack.

2. (Amended) [A] The method as claimed in claim 1 further comprising:

[associating] assigning a local status [with] to the object;

changing the status of the object to global under certain conditions.

3. (Amended) [A] The method as claimed in claim 2 further comprising deleting from [the] a given thread heap one or more local objects when they are not [reachable] accessible from a local root.

4. (Amended) [A] The method as claimed in claim 3 where [reachability] accessibility is determined by tracing from the local root.

5. (Amended) [A] The method as claimed in claim 4 wherein the status of an object in the given thread heap is changed to global if the object is assigned to a static variable or if the object is assigned to a field in any other object.

6. (Amended) [A] The method as claimed in claim 3 further comprising intercepting assignment operations to an object in [a] the thread heap to assess whether the object status should be changed.

7. (Amended) [A] The method as claimed in claim 6 wherein the multithreaded processing environment is a virtual machine.

8. (Amended) [A] The method as claimed in claim 7 wherein the virtual machine comprises an interpreter and the write operation code in the interpreter is modified to perform the checking of assignment of the object.

9. (Amended) [A] The method as claimed in claim 8 wherein the virtual machine comprises a just-in-time compiler, and wherein native compiled write operation code includes native code to perform the checking of assignment of the object.

10. (Amended) [A] The method as claimed in claim 9 further comprising using spare capacity in the object header for [the] a flag.

11. (Amended) [A] The method as claimed in claim 10 further comprising using multiples of 2 or more bytes in a thread heap to store the objects whereby there is at least one spare bit in the object length variable and using the at least one spare bit as the flag.

12. (Amended) [A] The method as claimed in claim 11 further comprising moving objects whose status is global from the thread heap to a global heap.

13. (Amended) [A] The method as claimed in claim 12 further comprising compacting the [reachable] accessible local objects in a thread heap.

14. (Amended) [A] The method as claimed in claim 1 wherein certain objects are associated with a global status on creation.

15. (Amended) [A] The method as claimed in claim 14 where said certain objects include Class objects, Thread objects and Runnable objects.

16. (Amended) [A] The method as claimed in claim 14 further comprising [the] a step of analysing whether an object is likely to be made global and associating such an object with a global status on creation.

17. (Amended) [A] The method as claimed in claim 16 further comprising allocating objects assigned as global on creation to the global heap.

18. (Amended) A system for managing memory in a multi-threaded processing environment comprising:

respective local thread stacks and heaps;

a global heap;

means for creating an object in a thread heap; and means for monitoring the object to determine whether the object is [a local root] referenced only from a given thread stack.

19. (Amended) [A] The system as claimed in claim 18 further comprising means for associating a local status with the object and means for changing the status of the object to global under certain conditions.

20. (Amended) [A] The system as claimed in claim 19 further comprising means for deleting from the thread heap one or more local objects when they are not reachable from a local root.

21. (Amended) [A] The system as claimed in claim 20 further comprising:

means for changing the status of an object in the thread heap to global if the object is assigned to a static variable or if the object is assigned to a field in any other object.

22. (Amended) A computer program product stored on a computer readable storage medium for, when executed on a computer, managing memory in a multi-threaded processing environment including respective local thread stacks and heaps and a global heap, said product comprising:

[means] instructions for creating an object in a thread heap; and

[means] instructions for monitoring whether the object is [a local root] referenced only from a given thread stack.

23. (Amended) [A] The product as claimed in claim 22 further comprising:

means for associating a local status with the object;

means for changing the status of the object to global under certain conditions.

24. (Amended) [A] The product as claimed in claim 23 further comprising means for deleting from the thread heap one or more local objects when they are not a local root.

25. (Amended) [A] <u>The</u> product as claimed in claim 24 where [reachability] <u>accessibility</u> is determined by tracing from the iocal root.

26. (Amended) [A] <u>The</u> product as claimed in claim 25 wherein the status of an object in the thread heap is changed to global if the object is assigned to a static variable or if the object is assigned to a field in any other object.

27. (Amended) [A] <u>The</u> method as claimed in claim 4 wherein the status of an object in the thread heap is changed to global if the object is assigned to a static variable or if the object is assigned to a field in a global object.

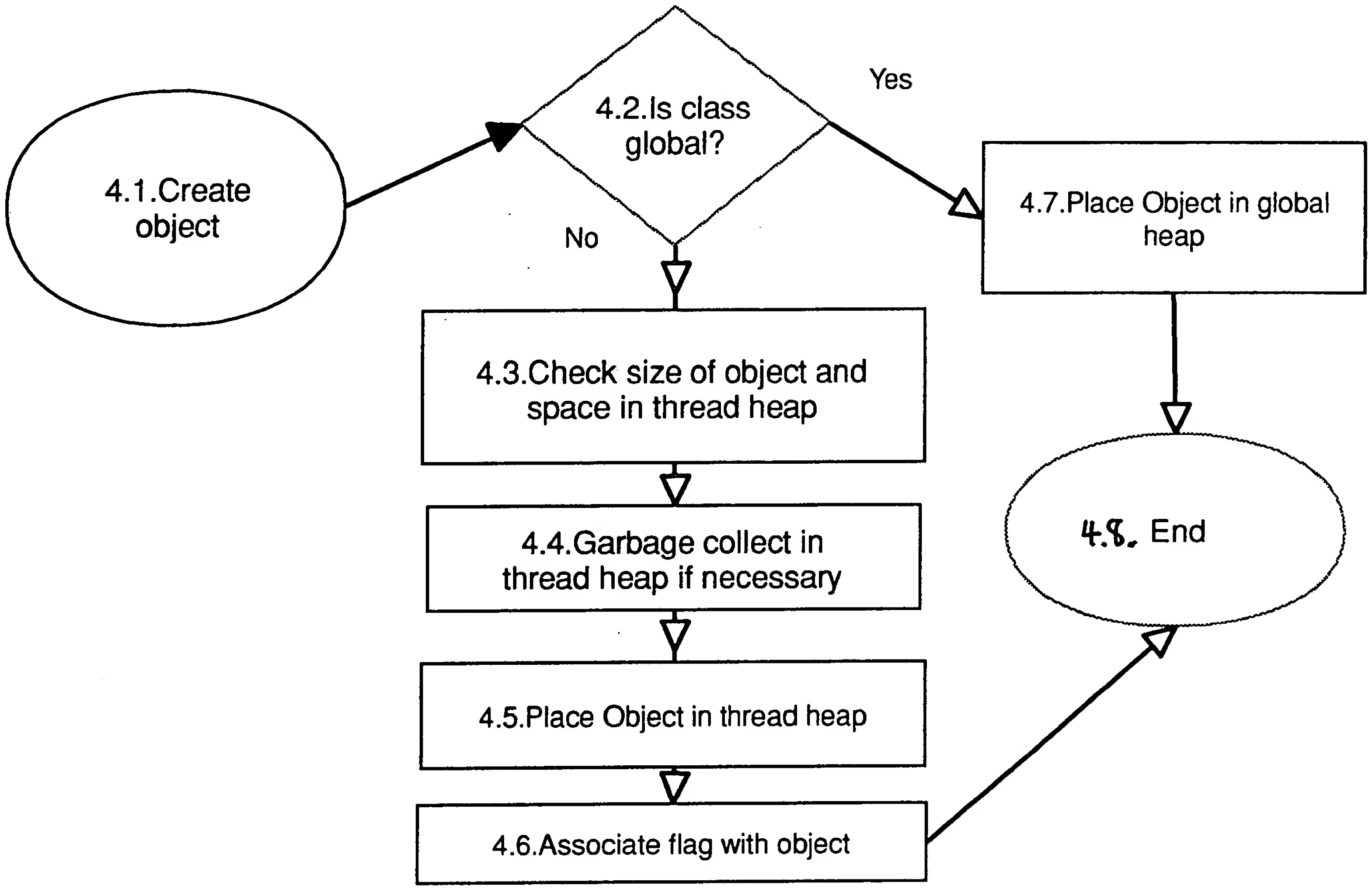
4.1.Create object
4.2.Is class global?
Yes
No
4.7.Place Object in global heap
4.3.Check size of object and space in thread heap
4.4.Garbage collect in thread heap if necessary
4.5.Place Object in thread heap
4.6.Associate flag with object
4.8. End

Figure 4